Automated Assume-Guarantee Reasoning for Omega-Regular Systems and Specifications

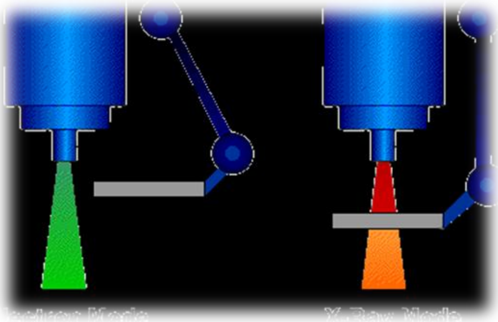
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Arie Gurfinkel and Sagar Chaki
April 13, 2010
Second NASA Formal Methods Symposium

When Failure is Not an Option

Failure is not an option for

- Safety Critical Systems (e.g., X-rays machines)
- Medical Devices (e.g., infusion pumps, …)
- Embedded Software (e.g., cars, airplanes)
- Security Vulnerabilities (e.g., nuclear plants)
- …



Formal software verification is essential to guarantee absence of failures

- Automated techniques include model checking and static analysis …
- which can be used to validate, for example, safety and security, behavior prior to program execution
- and provide objective evidence of safe behavior

Assume Guarantee Reasoning

System M_1 in parallel with system M_2 satisfies specification S

iff there exists an *assumption* A such that

- M_1 in parallel with A satisfies S
- M_2 satisfies the assumption A

$$\frac{M_1 \parallel A \models S \qquad M_2 \models A}{M_1 \parallel M_2 \models S}$$

How to *automatically* find a sufficiently good assumption?!

Related Work

Safety properties

- Giannakopoulou et al. ASE 2002 – computing weakest assumption
- Cobliegh et al. TACAS 2003 – using L* to learn “good-enough” assumption
- Barringer et al. SAVCBS 2003 – AG proof rules, soundness, completeness
- many follow up works to improve algorithms, complexity, applicability, etc.

Liveness properties

- Farzan et al. TACAS 2008 – $L^{\$}$ a learning algorithm for omega-regular languages

- **THIS PAPER**
  - Assume-Guarantee proof rules for reactive (omega-regular) systems
  - Soundness and (in)completeness
  - Two new learning algorithms for infinitary (finite + infinite) languages
  - A unifying framework for learning-based automated AG (see paper)

Outline

Background

- Model of concurrency: LTS, composition, specifications, etc.
- Active learning of regular languages: L^*
- Learning-based Automated Assume Guarantee Framework

Non-Circular Assume Guarantee Rule (AG-NC)

- Soundness and *in-completeness* for omega-regular languages
- Soundness and *completeness* for ∞-regular languages
- Learning algorithms for ∞-regular languages

Circular Assume Guarantee Rule (AG-C)

- Soundness and completeness

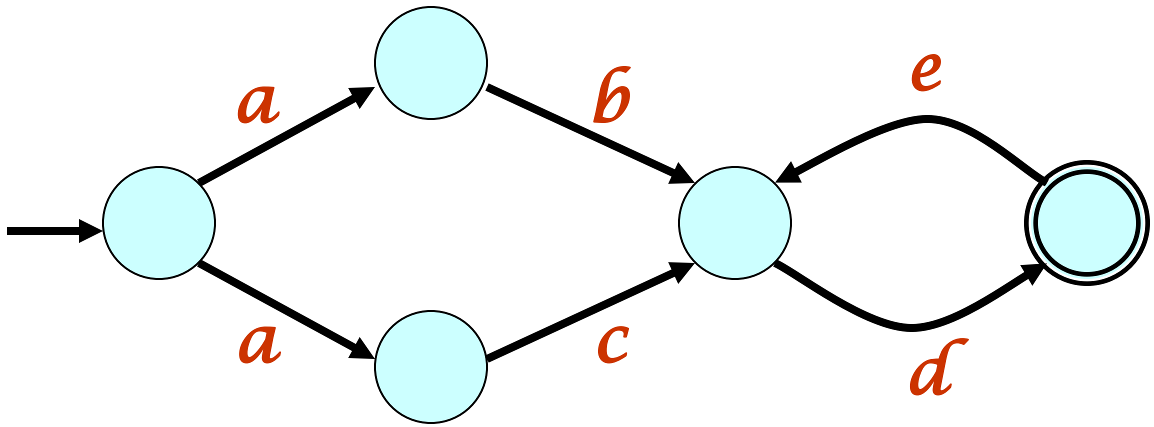
Conclusion and Future Work

Labeled Transition System (LTS)

M = (Q, I, Σ, T)

- Q -- non-empty set of states
- $I \in Q$ an initial state
- Σ -- set of actions (a.k.a, the alphabet)
- $T \subseteq Q \times \Sigma \times Q$ – a transition relation

+ **FA or Buchi acceptance condition**



$\mathcal{M}$

$\Sigma(\mathcal{M}) = \{a,b,c,d,e,f\}$

Operational Semantics

CSP Semantics

- handshake (synchronize) over shared actions
- otherwise, proceed independently (asynchronously)

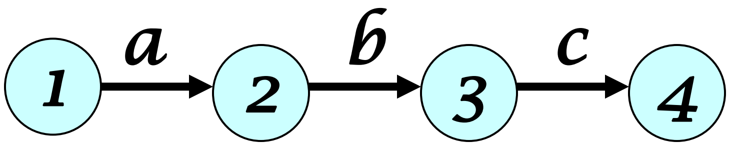
Composition $M_1 \,||\, M_2$ is

- State of $M_1 \,||\, M_2$ is of the form (s_1,s_2), where s_i is a state of M_i

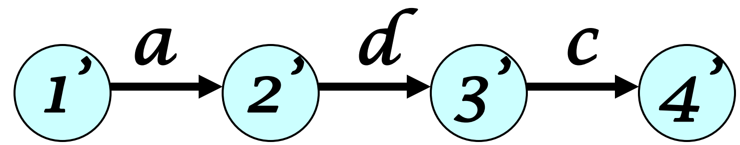
$$\frac{s_1 \xrightarrow{a} t_1 \qquad a \notin \Sigma(\mathcal{M}_2)}{(s_1,s_2) \xrightarrow{a} (t_1,s_2)} \qquad \frac{s_2 \xrightarrow{a} t_2 \qquad a \notin \Sigma(\mathcal{M}_1)}{(s_1,s_2) \xrightarrow{a} (s_1,t_2)}$$

$$\frac{s_1 \xrightarrow{a} t_1 \qquad s_2 \xrightarrow{a} t_2}{(s_1,s_2) \xrightarrow{a} (t_1,t_2)}$$

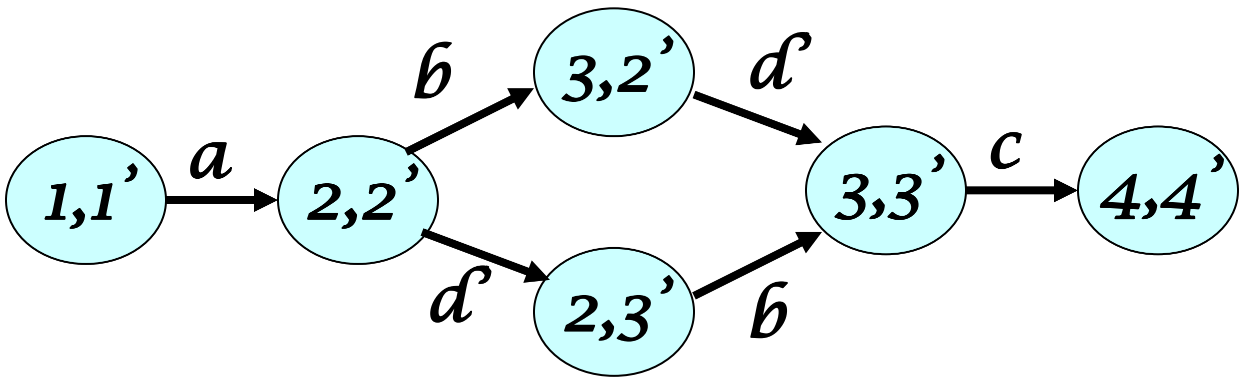
Example of Composition



$\mathcal{M}_1 \quad \Sigma = \{a,b,c\}$



$\mathcal{M}_2 \quad \Sigma = \{a,d,c\}$



$\mathcal{M}_1 \parallel \mathcal{M}_2 \ \Sigma = \{a,b,d,c\}$

L* (Angluin 1987, Rivest & Schapire 1993)

L* learner

$\Sigma(U)$ ⇨

Minimally Adequate Teacher (MAT)

U -- regular language

Membership Query

$s \in U$?

yes/no

Candidate Query

$\mathcal{L}(A) = U$?

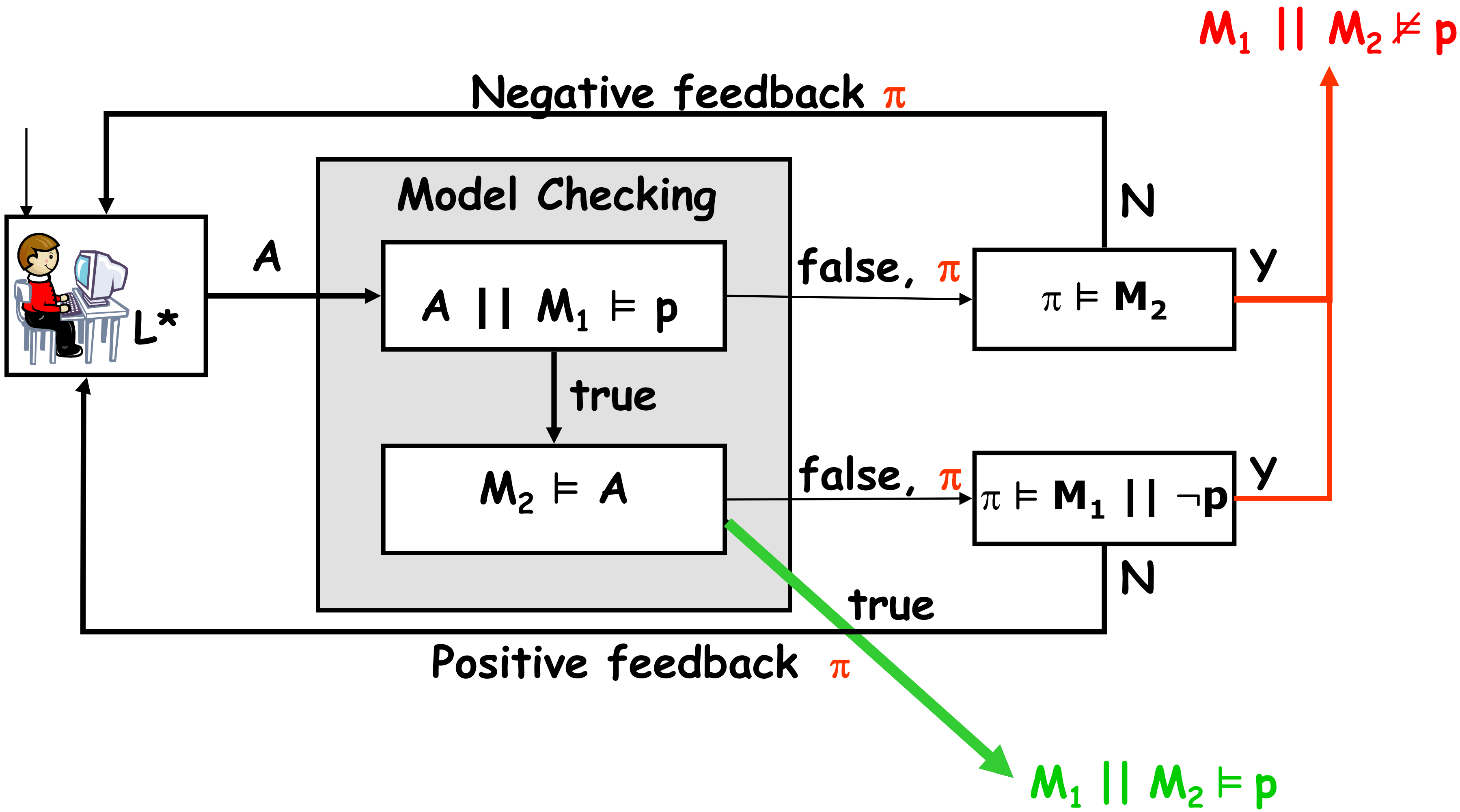
No. $\pi \in U \,\Delta\, \mathcal{L}(A)$

yes. $\mathcal{L}(A) = U$

DFA A

$\mathcal{L}(A) = U$

Assume Guarantee with Learning



Outline

Background

- Model of concurrency: LTS, composition, specifications, etc.
- Active learning of regular languages: L^*
- Learning-based Automated Assume Guarantee Framework

Non-Circular Assume Guarantee Rule (AG-NC)

- Soundness and *in-completeness* for omega-regular languages
- Soundness and *completeness* for ∞-regular languages
- Learning algorithms for ∞-regular languages

Circular Assume Guarantee Rule (AG-C)

- Soundness and completeness

Conclusion and Future Work

AG-NC: Non-Circular Assume Guarantee Rule

$$L \preceq S \text{ iff } L{\downharpoonright}\Sigma_S \subseteq S$$

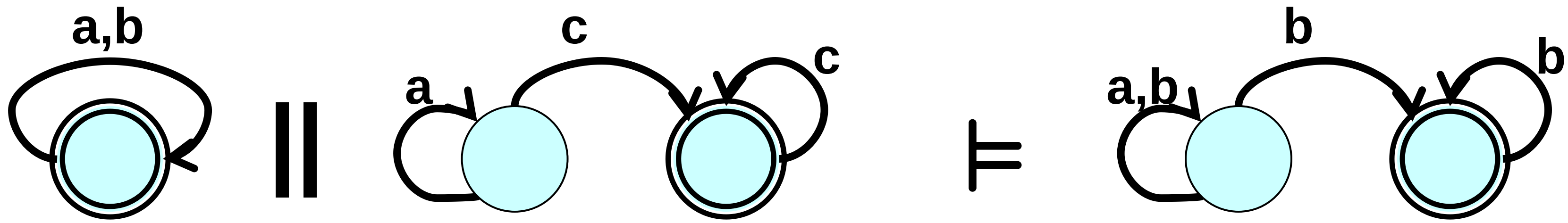
$$\frac{(L_1 \,\|\, L_A) \preceq L_S \qquad L_2 \preceq L_A}{(L_1 \,\|\, L_2) \preceq L_S}$$

$$\Sigma_A = (\Sigma_1 \cup \Sigma_S) \cap \Sigma_2$$

Complete for Safety (regular) properties (Σ^*)

Incomplete for Liveness (omega-regular) properties (Σ^ω)

Proof of Incompleteness (by Counterexample)



Σ_1 = {a, b} Σ_2 = {a, c} Σ_S = {a, b}

$L_1 = (a+b)^\omega$ $L_2 = a^*c^\omega$ $L_S = (a+b)^*b^\omega$

Assumption alphabet: Σ_A = {a}

BUT, there is no assumption $L_A \subseteq \Sigma_A^\omega$ to apply AG-NC

$A_1 = \emptyset \quad L_2 \not\preceq A_1$ $A_2 = a^\omega \quad L_1 \| A_2 \not\preceq L_S$

AG-NC: Infinite Trace Containment

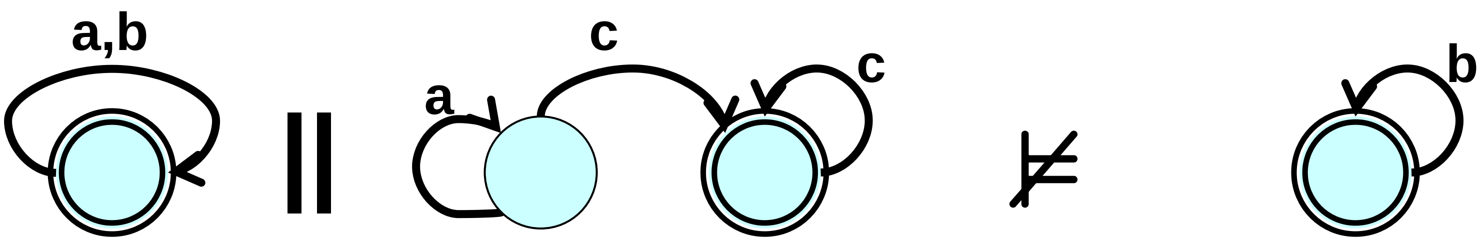
$\mathbf{L} \preceq_\omega \mathbf{S}$ **iff** $\omega(\mathbf{L}\downharpoonright\Sigma_S) \subseteq \omega(\mathbf{S})$

$$\frac{(L_1 \,||\, L_A) \preceq_\omega L_S \qquad L_2 \preceq_\omega L_A}{(L_1 \,||\, L_2) \preceq_\omega L_S}$$

$\Sigma_A = (\Sigma_1 \cup \Sigma_S) \cap \Sigma_2$

NOT SOUND!

Proof of Unsoundness (by Counterexample)



Σ_1 **= {a, b}** Σ_2 **= {a, c}** Σ_S **= {a, b}**

$L_1 = (a+b)^\omega$ $L_2 = a^*c^\omega$ $L_S = b^\omega$

Σ_A **= {a}**

BUT, $L_A = \emptyset$ satisfies all premises of (modified) AG-NC

AG-NC: Relaxing Assumption Alphabet

$$\frac{(L_1 \parallel L_A) \preceq L_S \qquad\qquad L_2 \preceq L_A}{(L_1 \parallel L_2) \preceq L_S}$$

$\cancel{\Sigma_A = \Sigma_1 \; \Sigma \cap \Sigma_2}$

Complete for Safety (regular) properties (Σ^*)

Complete for Liveness (omega-regular) properties (Σ^ω)

Assumption "knows" about internal actions of L_1 and L_2

Not "truly" compositional

AG-NC: Restoring Completeness

Theorem: Let L_1 and L_S be two languages, and Σ_A an alphabet s.t. $\Sigma_1 \cup \Sigma_A = \Sigma_1 \cup \Sigma_S$. Then, $L_A = \mathcal{C}((L_1 \,||\, \mathcal{C}(L_S))\lfloor\Sigma_A)$ is the *weakest* assumption such that $L_1 \,||\, L_A \preceq L_S$

Corollaries:

AG-NC is complete for any class of languages closed under *projection* and *complement*

AG-NC is complete for Σ^*

AG-NC is complete for $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$

Need a learning algorithm for Σ^∞!

Learning Infinitary Language U: Approach 1

Use L* and L$^{\$}$ *simultaneously* to learn a DFA *D* and a BA *B* such that $\mathcal{L}(\text{DFA}) = {}^*(U)$ and $\mathcal{L}(\text{BA}) = \omega(U)$

Assume M is a MAT for U

Use M to answer membership queries until *both* learners generate a candidate query

Use M to verify the candidate query: $\mathcal{L}(D) \cup \mathcal{L}(B) = U$

- on success, stop
- if counterexample is finite, send to L* and resume until next DFA candidate
- if counterexample is infinite, send to L$^{\$}$ and resume until next BA candidate

Two learners. Possibly a lot of redundancy.

Learning Infinitary Language U: Approach 2

Use L$^{\$}$ to learn U.τ^ω, where τ is a "fresh" symbol not in Σ_U

Assume M is a MAT for U

To answer a membership query infinite word s

- if s = t.τ^ω and t $\in \Sigma_U^\infty$ then ask M whether t $\in$ U and forward answer back
- Otherwise, answer "no"

To answer a candidate query with candidate BA *C*

- if $\mathcal{L}$(C) $\nsubseteq \Sigma_U^\infty.\tau^\omega$ return $\pi \in \mathcal{L}$(C) $\setminus \Sigma_U^\infty.\tau^\omega$
- otherwise, forward candidate query *($\mathcal{L}$(A)$\downharpoonleft\Sigma$), ω($\mathcal{L}$(A)$\downharpoonleft\Sigma$) to M

Single learner, BUT larger alphabet

Circular AG-rule (AG-C): Summary

$$\Sigma_{A1} = \Sigma_{A2} = (\Sigma_1 \cap \Sigma_2) \cup \Sigma_S$$

$$\frac{L_1 \parallel L_{A1} \preceq L_S \qquad L_2 \parallel L_{A2} \preceq L_S \qquad \mathcal{C}(L_{A1}) \parallel \mathcal{C}(L_{A2}) \preceq L_S}{L_1 \parallel L_2 \preceq L_S}$$

Complete for Safety (regular) properties (Σ^{∞})

Complete for Liveness (omega-regular) properties (Σ^{ω})

BUT need to learn 2 assumptions AND assumption alphabet is larger

Learning-Based AG (LAG) Framework

Conformance	Rule	$\mathcal{A}$	Learner(s)	Oracle(s)	Checker
Regular Trace Containment	**AG-NC** [1]	DFA	$P_1 = P(\mathbf{L}^*)$	$Q_1 = Q(L_1, L_S, \Sigma_{NC})$	$V_{NC}(L_1, L_2, L_S)$
Regular Trace Containment	**AG-C** [2]	DFA	$P_1 = P_2 = P(\mathbf{L}^*)$	$Q_1 = Q(L_1, L_S, \Sigma_C)$ $Q_2 = Q(L_2, L_S, \Sigma_C)$	$V_C(L_1, L_2, L_S)$
∞-regular Trace Containment	**AG-NC**	DFA $\times$ BA	$P_1 = P(\mathbf{L})$	$Q_1 = Q(L_1, L_S, \Sigma_{NC})$	$V_{NC}(L_1, L_2, L_S)$
∞-regular Trace Containment	**AG-C**	DFA $\times$ BA	$P_1 = P_2 = P(\mathbf{L})$	$Q_1 = Q(L_1, L_S, \Sigma_C)$ $Q_2 = Q(L_2, L_S, \Sigma_C)$	$V_C(L_1, L_2, L_S)$
ω-regular Trace Containment	**AG-NC**	DFA $\times$ BA	$P_1 = P(\mathbf{L})$	$Q_1 = Q(L_1, L_S, \Sigma_{NC})$	$V_{NC}(L_1, L_2, L_S)$
ω-regular Trace Containment	**AG-C**	BA	$P_1 = P_2 = P(\mathbf{L}^\omega)$	$Q_1 = Q(L_1, L_S, \Sigma_C)$ $Q_2 = Q(L_2, L_S, \Sigma_C)$	$V_C(L_1, L_2, L_S)$

[1] Cobleigh, Giannakopoulou, Pasareanu, TACAS'03

[2] Barringer, Giannakopoulou, Pasareanu, SAVCBS'03

The last four rows are contributions of THIS PAPER

NEW!

Conclusion and Future Work

Compositional approach to verification is fundamental for scalability!

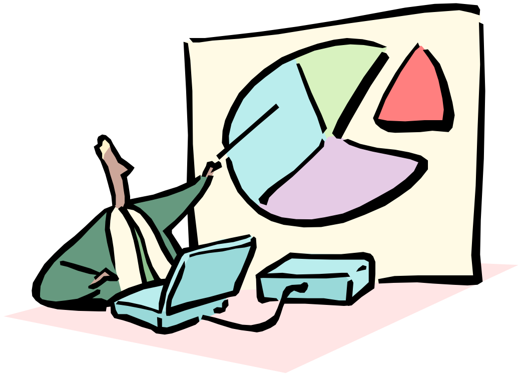
Automated AG for Liveness (omega-regular) properties

- Non-Circular Rule: soundness, (in)completeness
- Circular rule – remains sound and complete
- Two new learning algorithms for infinitary languages

Unified Framework for Learning-based Assume Guarantee Reasoning

Future Work

- implementation and empirical evaluation
- experiments with other learning algorithms



THE END

Contact Information

Presenter
Arie Gurfinkel
RTSS
Telephone: +1 412-268-5800
Email: arie@cmu.edu

U.S. mail:
Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

Web:
www.sei.cmu.edu
http://www.sei.cmu.edu/contact.cfm

Customer Relations
Email: info@sei.cmu.edu
Telephone: +1 412-268-5800
SEI Phone: +1 412-268-5800
SEI Fax: +1 412-268-6257